Solutions

2. (d) False. The counterexample is \( A = \{0^k : k \geq 0 \} \). First, we will show that \( A \) is not regular with a proof by contradiction. Suppose that \( A \) is regular and let \( p \) be its pumping length. By definition \( 2^{p+1} \in A \) and we can apply the pumping lemma to it since \( 2^{p+1} > p \) for every positive integer \( p \). By pumping this string up, we would get that \( 0^{2^{p+1}+x} \in A \) for some \( 0 < x \leq p \). Then, since
\[
2^{p+1} < 2^{p+1} + x \leq 2^{p+1} + p < 2^{p+1} + 2^{p+1} = 2^{p+2}
\]
we have that \( 0^{2^{p+1}+x} \notin A \) because \( 2^{p+1} + x \) cannot be written as a power of 2, reaching a contradiction and showing that \( A \) is not regular.

Finally, note that \( A^* = 0^* \) which is regular. The reason is that every non-integer \( n \) can be written in binary or as the sum of powers of 2. Thus, \( 0^n \) can be written as the concatenation of elements of \( A \) for every \( n \), as desired.

Note: The same proof can be used to show that \( A^* \text{ CFL} \) does not imply \( A \text{ CFL} \).

5. (a) First, we will show it is not regular with a proof by contradiction. Suppose this language is regular and has a pumping length \( p \). Then, by applying the pumping lemma to \( 0^p12^p0^p \) we can obtain a resulting string is not in language, as desired.

Finally, for proving that this language is a CFL, notice that all we need to show is that \( \{ w : w = w^R \} \) is a CFL because \( \{ w : |w| \text{ divisible by 4} \} \) is a regular language and the intersection of a CFL and a regular language is a CFL. However, for simplicity we will prove that \( \{ w : w = w^R \text{ and } |w| \text{ is even} \} \) is a CFL which should still yield the same result because its intersection with \( \{ w : |w| \text{ divisible by 4} \} \) gives us the same language we wanted to prove is a CFL. To do this, consider the following PDA. On input \( w \):

- Read and push symbols from the input tape to the stack until non-deterministically guessing when to stop.
- When decided to stop pushing, start popping symbols from stack and check that they match with the symbols from the input tape. If there is a mismatch or the stack gets empty while still there are more symbols to read, reject.
- When finishing reading the input tape, accept if stack is empty else reject.

which recognizes \( \{ w : w = w^R \text{ and } |w| \text{ is even} \} \) as desired.

(c) It is not regular. This can be proven with a proof by contradiction and applying the pumping lemma to \( 0^p1^p \) by pumping it up to get a string that is not in the language. However, it is a CFL. We can build a PDA that uses its stack to count how many more 1’s than 0’s we have at a given time. More specifically, on input \( w \):

- For every 0 that is read from the input tape:
  - If stack is empty, push the symbol \([-]\).
– If the top of the stack is a [−], push another [−] symbol.
– If the top of the stack is a [+] symbol.

• For every 1 that is read from the input tape:
  – If stack is empty, push the symbol [+].
  – If the top of the stack is a [+] symbol, push another [+] symbol.
  – If the top of the stack is a [−], pop it.
• When finishing reading the input:
  – If the stack is empty, accept.
  – If the top of the stack has a [+] symbol, accept.
  – If the top of the stack has a [−] symbol, reject.

This PDA uses the stack for counting how many more 1’s than 0’s it has seen so far. If the stack is empty, that means that there is an equal amount, if the stack has \( k \) [+] symbols then there are \( k \) more 1’s than 0’s and if there are \( k \) [−]’s symbols there are \( k \) less 1’s than 0’s. Thus, this PDA recognizes the target language which proves it is a CFL.

(g) This language is not regular but it is a CFL. We can show it is not regular by proving that its complement is not regular. This can be achieved with a proof by contradiction. Note that if the complement is regular and has a pumping length \( p \), then we can apply the pumping lemma to \( 1^{p+3}0^{2p+7}1^{p+3} \) and it is not hard to verify that by pumping it up twice we can get something that is not in the language. The reason is that by pumping it up twice we get a string that is not a palindrome, has odd length and the last three symbols are the same.

To proof that it is a CFL, we only need to show that \( \{ w : w \neq w^R \} \) is a CFL. We will build a PDA that recognizes this language by non-deterministically guessing where there is a mismatch that prevents \( w = w^R \) from happening. The key observation is that a mismatch occurs when for some \( i \), the symbol \( i \) and the symbol \( |w| + 1 - i \) do not match. More specifically, on input \( w \):

• Push symbols onto the stack until guessing non-deterministically when to stop.
• Skip symbols from input tape until guessing non-deterministically when to stop.
• Start popping and comparing symbol from the stack and input tape.
• When finished reading the input tape, accept if a mismatch was found in the popping process and the stack ends up being empty. Otherwise, reject.

Note that we require that the stack is empty at the end to ensure that any mismatch corresponds to a mismatch of symbol at position \( i \) and \( |w| + 1 - i \). Thus, this PDA recognizes \( \{ w : w \neq w^R \} \), proving it is a CFL as desired.

7. (d) Let \( B = \{ \langle M \rangle : M \text{ is a TM that loops on some input} \} \).

– To show \( B \) is not co-recognizable, consider the following modification to \( f'_3 \) of \( f_3 \).
  \( f'_3(\langle M, w \rangle) = \langle N'_3 \rangle \), where \( N'_3 \) on input \( x \) it loops if \( x \) is an accepting computation history of \( M \) on \( w \) and accepts otherwise. Note that \( \langle M, w \rangle \in A_{TM} \iff N'_3 \) loops on an input, showing that \( A_{TM} \leq_m B \).
To show $B$ is not recognizable, consider the following modification to $f_1$ of $f_1. f'_1(⟨M, w⟩) = ⟨N'_1⟩$ where $N'_1$ on input $x$ ignores $x$ and simulates $M$ on $w$. If $M$ accepts, $N'_1$ accepts. If $M$ halts and rejects, $N'_1$ loops. Note that $⟨M, w⟩ \in A_{TM} \iff N'_1$ loops on some input. Thus, $A_{TM} \leq_m B$. 

- Let $C = \{⟨M⟩ : M$ is a TM that loops on all inputs$\}$. First we will show that its complement is recognizable. This can be done by a TM that on input $⟨M⟩$ it simulates $M$ on all inputs by using the dove-tailing technique. If $M$ halts on an input, this TM will accept. To show that $C$ is unrecognizable, just use the modified reduction $f'_1$. Note that actually, $⟨M, w⟩ \in \overline{A_{TM}} \iff N'_1$ loops on all inputs, showing that $\overline{A_{TM}} \leq_m C$.

- Let $B = \{⟨M, N⟩ : M, N$ are TMs and $\varepsilon \in L(M) \cup L(N)\}$. $B$ is recognizable. Its recognizer will run both $M$ and $N$ on $\varepsilon$ in parallel and accept if one of the two accepts.

  Consider the following modified reduction $f'_1$ of $f_1$. $f'_1(⟨M, w⟩) = ⟨N_1, N_2⟩$, where both $N_1$ and $N_2$ on input $x$, ignore $x$ and simulate $M$ on $w$. They only accept if $M$ accepts. Since $\varepsilon \in L(N_1), L(N_2) \iff ⟨M, w⟩$, we have that $A_{TM} \leq_m B$, meaning $B$ is undecidable.

- Let $C = \{⟨M, N⟩ : M, N$ are TMs and $\varepsilon \in L(M) \cap L(N)\}$. $C$ is recognizable. Its recognizer will again on input $⟨M, N⟩$, run $M$ and $N$ on input $\varepsilon$ in parallel and accept only if both accept. For showing that it is undecidable, note that the same reduction used for proving $B$ undecidable works.

- Let $D = \{⟨M, N⟩ : M, N$ are TMs and $\varepsilon \in L(M) \setminus L(N)\}$. $D$ is neither recognizable nor co-recognizable.

  - To show $D$ is not co-recognizable, consider the following reduction $g_1(⟨M, w⟩) = ⟨N_1, N_{\text{reject}}⟩$. $N_1$ on input $x$ ignores $x$ and runs $M$ on $w$, accepting only if $M$ accepts $w$. $N_{\text{reject}}$ rejects all inputs. Notice that $M$ accepts $w$ implies that $L(N_1) \setminus L(N_{\text{reject}}) = \Sigma^* \setminus \emptyset = \Sigma^*$ and $M$ rejects $w$ implies that $L(N_1) \setminus L(N_{\text{reject}}) = \emptyset \setminus \emptyset = \emptyset$. Thus, $⟨M, w⟩ \in A_{TM} \iff \varepsilon \in L(N_1) \setminus L(N_{\text{reject}})$, proving that $A_{TM} \leq_m D$.

  - To show $D$ is not recognizable, consider the following reduction $g_2(⟨M, w⟩) = ⟨N_{\text{accept}}, N_1⟩$. $N_1$ is defined as in the previous case. $N_{\text{accept}}$ accepts all inputs. Notice that $M$ accepts $w$ implies $L(N_{\text{accept}}) \setminus L(N_1) = \emptyset$ and $M$ rejects $w$ implies that $L(N_{\text{accept}}) \setminus L(N_1) = \Sigma^*$. Thus, $⟨M, w⟩ \in \overline{A_{TM}} \iff \varepsilon \in L(N_{\text{accept}}) \setminus L(N_1)$, showing that $\overline{A_{TM}} \leq_m D$.

- Let $E = \{⟨M, N⟩ : M, N$ are TMs and $L(M) \leq_m L(N)\}$. $E$ is neither recognizable nor co-recognizable. To show this, we will use the fact that $\Sigma^* \leq_m \Sigma^*, \emptyset \leq_m \emptyset$ and $\Sigma^* \not\leq_m \emptyset, \emptyset \not\leq_m \Sigma^*$.

  - To show it is not co-recognizable, consider the reduction $h_1(⟨M, w⟩) = ⟨N_1, N_{\text{accept}}⟩$. Then $M$ accepts $w$ implies that $L(N_1) = \Sigma^*$ and $M$ rejects $w$ implies that $L(N_1) = \emptyset$. Thus, $⟨M, w⟩ \in A_{TM} \iff L(N_1) \leq_m L(N_{\text{accept}})$, meaning that $A_{TM} \leq_m E$.

  - To show it is not recognizable, consider the reduction $h_2(⟨M, w⟩) \in A_{TM} = ⟨N_1, N_{\text{reject}}⟩$. Notice by a similar analysis we conclude that $\overline{A_{TM}} \leq E$.

- Let $B = \{⟨M, w⟩ : M$ is a TM which moves its head left on the leftmost cell on $w\}$.

(g)
First, notice that $B$ is recognizable. Its recognizer on input $\langle M, w \rangle$ will simulate $M$ on $w$ and accept if $M$ moves its head left on the leftmost cell. To show it is undecidable, consider the following reduction $g_1(\langle M, w \rangle) = \langle N, w \rangle$. $N$ on input $x$ will first shift the input $x$ to the right by one cell and add a special symbol $\$ in the leftmost cell. Then, it will proceed to simulate $M$ on $x$. If $M$ accepts, it will move its head to the left of the leftmost cell. Notice that if $M$ does not accept, it will never do that because of the special symbol that we put in the leftmost cell. Thus, $\langle M, w \rangle \in A_{TM} \iff \langle N, w \rangle \in B$, showing that $B$ is undecidable.

• Suppose a machine runs for a long long time and it has not turned its head left (long enough that it is way outside of the portion of the tape that contained the input). Note that all this machine is doing at this point is reading a blank symbol, writing something in its place and moving its head right. From the machine’s local point of view, all it sees is a state and a blank symbol. Thus, if it runs for a while it will get to a point where it has been at the same state twice after reading blank symbols and moving its head right. If this happens, the machine will just enter an infinite loop. More formally, here is description of the decider for this language:

On input $\langle M, w \rangle$:
- Simulate $M$ on $w$ for $|w| + |Q| + 1$ steps.
- If $M$ has not turned its head left, reject. (It has entered a loop as explained above).
- Else, accept.

The $|w|$ factor is to ensure that $M$ reaches the blank symbols portion of the tape. Then, if $M$ read blank symbols and move right for $|Q| + 1$ steps, by the pingeonhole principle $M$ will enter the same state twice, meaning that it will loop forever.

**Obs:** What it might be weird at first is that are we not considering the tape contents when ”counting configurations” here. The reason is that by always moving the head right, the TM never interacts with what it wrote. However, it is important for the normal because the TM can go back to what it wrote and use it.